

UNITED STATES PATENT APPLICATION

FOR

**METHOD AND APPARATUS FOR AUTOMATIC LOAD-BALANCING ON  
MULTISEGMENT DEVICES**

BY

KAREN E. KIMBALL  
ROBERT L. FAULK, JR.  
ROBERT M. MCGUIRE

20070525001

# METHOD AND APPARATUS FOR AUTOMATIC LOAD-BALANCING ON MULTISEGMENT DEVICES

5

## BACKGROUND OF THE INVENTION

10

### TECHNICAL FIELD

The invention relates to electronic communications networks. More particularly, the invention relates to the automatic configuration of hardware and software networking products for network segmentation to balance the flow of communications and data between nodes on such networks.

5

### DESCRIPTION OF THE PRIOR ART

A network segment consists of repeaters, end stations, and connectivity between the stations (for example, see standard ISO/IEC 8802-3 for information on Ethernet networks). A set of predefined rules for communicating, called "protocols," determines how and when end-nodes on the network segment are allowed to communicate with one another, and what form such communication takes.

20

25

In a data communications network, data are transferred in the form of packets which contain the address of the sending station (the source address) and the address of the intended recipient (the destination address). The exact packet format, and the predefined rules for interpreting the contents of such packets, are again specific to the network type and are made up of several levels of protocols. As described herein, a network

30

segment has the capacity to transfer a limited amount of data per second. When more than the maximum media speed of network data transfer capacity is required, multiple segments can be used. These segments can be interconnected through the use of bridges or other packet-forwarding devices, which allow communication between different segments (bridges are defined in standard ISO/IEC 10038).

The term switch has recently been applied to bridges. Switches are fast bridges which generally do not use a CPU to process the packets. Bridges contain two or more ports, each of which connects to a segment. When two stations on different segments communicate with each other, the bridge forwards the packets between the two segments. When the stations are on the same segment, the bridge does not forward the packets to any other segment. The bridge may buffer data received on its ports to allow forwarding of the data later onto a different segment that was busy when the transmission first occurred.

Over time, a network may grow as more users (end-nodes) are added. Each of these new end-nodes must then compete with existing end-nodes for access to the network. As the number of end-nodes which might potentially access the network medium grows, it becomes statistically more likely that the medium is in use when a given end-node wants to transmit, thereby requiring the end-node to wait. In addition, available network resources may be constrained when just a few end-nodes have heavy resource impact. Nodes which transmit a lot of packets, which transmit a lot of broadcast or multicast packets (received by all or many stations), or which create a lot of network disruptions (e.g., collisions, late collisions, etc.), consume a lot of the available network resources, leaving less for other end-nodes. Thus, schemes which increase the amount of network resources available (*i.e.* the amount of potential accessibility to the network medium), without changing

the type of physical network used (preserving the users' investment), are of great use and interest.

When the network medium is shared (as opposed to switched), one way of increasing the amount of available network resources is to split the shared medium into multiple segments (i.e., segment the network). This limits the number of users who have access to any one of these segments. If it is desired to have end-nodes in a segment communicate with end-nodes in other segments, bridging or switching or routing across the segments may be employed, so that network packets for devices that are not in the immediate segment are forwarded until they reach the segment of the destination end-node.

While segmenting the network medium can improve resource availability, the choice of where to segment, e.g. which nodes should be assigned to which segment, affects the success of the results. If all of the nodes which impact network resources the most remain on the same segment, access for any one of those nodes is not much improved by this scheme, while the less resource-impacting nodes on other segments have greatly increased opportunity for access but far less need of it. In this situation, the increased resources are wasted on the wrong end-nodes.

Additionally, network systems are inherently complex. A significant amount of technical education is required to understand, configure, and maintain any particular type of physical network, e.g. 10Base-T, Ethernet, Token Ring, or FDDI networks. As the use of networks becomes widespread, the number of users responsible for such networks grows, while those among them who have sufficient knowledge to accomplish these tasks declines. Thus, schemes which can configure, maintain, or improve network conditions without requiring much specific knowledge on the part of the user are also of great use and interest. Such ease-of-use, i.e. user-friendly, schemes are

often encompassed in software which is either embedded in network devices or external to them. These software applications generally are known as intelligent network agents.

One network products vendor is presently marketing a 10Base-T network switch that allows any of its ports to be assigned to any one of four internal backplane segments. The product is described as having a load-balancing feature which can be activated in one of several ways:

- On-demand by the user (*i.e.* a manual trigger);
- At a designated time or interval; or
- When a user-defined threshold is reached (this is a threshold of a network value defined in the product's Management Information Base (MIB) and configured via the Simple Network Management Protocol (SNMP). For example, the MIB value could be a collision count, a segment utilization pattern, or a switch utilization pattern.

These mechanisms for triggering such actions are not new. The first two have existed in practice for some time, and the third is an aspect of recent implementations of the Remote MONitoring (RMON) suite of features. The algorithms that are actually used by the product's load-balancing feature to determine which ports to place on which of the four internal backplane segments have been informally described as being based on end-node communication patterns gleaned from address-pair information in the switch's address table. Thus, this approach requires that the switching fabric and intelligent network agent components are fully integrated into the intelligent agent's hardware environment.

Accordingly, the state of the art provides an automatic load-balancing solution that does not accommodate an inability on the part of the user or agent to communicate with the switching fabric (e.g., packet-forwarding, address table, etc.) hardware. This arrangement has a major disadvantage in that it is unusable with switches and other packet-forwarding devices which are fully external to the intelligent agent's hardware environment. It would therefore be advantageous to provide an agent that can intelligently improve network resource availability, without requiring special, integrated packet-forwarding hardware.

In addition, the prior method may result in unnecessary load-balancing. It would therefore be advantageous to reduce network disruption by performing such load-balancing only when it actually improves network resource availability.

Furthermore, it is advantageous to provide an agent that can intelligently improve network resource availability while requiring only a simple activate action on the part of the user, such that no further action or special knowledge of the agent, its environment, or networks in general (such as how to set a threshold level, or even which threshold values would be meaningful) is required on the part of the user.

### **SUMMARY OF THE INVENTION**

The invention provides an intelligent network agent that implements a feature referred to as automatic load-balancing. The invention addresses both of the major issues of network resource impact and ease-of-use described above, and intelligently improves network throughput, while requiring only a simple activate action on the part of the user such that no further action or special knowledge of the agent, its environment, or networks

in general is required on the part of the user. The invention is designed to minimize network disruption by load-balancing only when it actually improves the network resource availability. Furthermore, because of the ability to use this invention in environments where packet-forwarding support hardware is external to the agent's environment, the invention works in situations where prior inventions do not, including cases where an external switch or other packet-forwarding device is manufactured by another vendor.

The automatic load-balancing method and apparatus further comprises three major components:

- A decision-making process for automatically determining when a given load-balancing activation currently would not be beneficial;
- An intelligent load-balancing port-distribution algorithm for assigning ports to segments; and
- A feature which allows undoing a prior load-balancing.

One goal for an intelligent feature of any sort involves determining when to actually implement the feature. The herein disclosed automatic load-balancing invention checks for situations in which a given automatic load-balancing activation should not occur by weighing the potential benefit of the action against its consequences, and by making an intelligent decision for the user while preventing needless network disruption.

Automatic load-balancing moves end-nodes (or groups of end-nodes) after the system has been running to increase the overall throughput of the network. This aspect of the invention redistributes hub ports to segments with the goal of evenly balancing resource load across all of the known segments. Past history is used to anticipate the future load, *i.e.* history since

either activation of the intelligent agent or since the last time the load-balancing feature itself was activated.

The feature to undo a prior load-balancing takes information stored at a prior load-balancing activation and uses it to return ports to the previous configuration.

### **BRIEF DESCRIPTION OF THE DRAWINGS**

Fig. 1 is a block schematic diagram of an electronic communications network;

Fig. 2 is a block schematic diagram of a preferred hardware environment according to the invention;

Fig. 3 is a block schematic diagram of an apparatus for automatic load balancing on segmented devices according to the invention; and

Fig. 4 is a flow diagram showing a preferred implementation of an intelligent port distribution algorithm according to the invention.

### **DETAILED DESCRIPTION OF THE INVENTION**

Fig. 1 is a block schematic representation of a typical electronic communications network, such as an Ethernet network. The network consists of several Local Area Networks (LANs) 12-16, each of which is interconnected through a number of bridges 17-21. Each LAN is connected to one or more bridges. Connection between a LAN and a bridge is through a port. For example, ports 211, 212, 213 are shown in connection with the bridge 21. The principle functions of the bridge are to relay and filter data frames, maintain the information required to make frame filtering and



relaying decisions, and management of the foregoing operations. Although the preferred embodiment of the invention is discussed in connection with an Ethernet network, the invention can readily operate with networks other than Ethernet networks. In fact, the invention herein described can apply at least to FDDI, as well as any of the networks specified in the IEEE 802 family of networking standards (e.g. 802.3 and 802.5).

The invention relates to the automatic configuration of hardware and software networking products for network segmentation, e.g. with regard to balancing the flow of communications and data between nodes on network segments to obtain more throughput on the network. Such automatic configuration off-loads responsibility for the management of the network and network devices from the system administrator, thereby saving time and training. Automatic configuration as disclosed herein refers to a decision made by the load balancing system including, potentially, an out-of-the-box first plug-in decision, and any adjustment to the network devices after the network system is in use (e.g. a "find new load-distribution," "set daytime config," or a "set off-hours config" decision).

Fig. 2 is a block schematic diagram of a preferred hardware environment according to the invention. The hardware environment consists of a network hub 34 (or connected stack of hubs 35, 36, 37), a network management card 30 which contains the processor and firmware required for the intelligent network agent features described herein, and an optional internal packet-forwarding module 32 or external packet-forwarding module 38 required to maintain communication across all four of the hub stack's internal segments 1... M.

The preferred packet-forwarding module 32 is integrated into the system only insofar as the module itself can be detected by the intelligent network agent 30. The hub 34 (or connected stack of hubs 35, 36, 37) has backplane

connections 1..a, (a+1)..b, (y+1)..z which link the external ports 1..a, (a+1) ..b, (y+1) ..z to any one of the hub stack's backplane segments 1..M. The agent and the packet-forwarding module's backplane connections tie directly into the hub stack's backplane segments 1..M in the preferred embodiment of the invention. Other implementations of the invention can be applied wherein a fully external packet forwarding module 38 connects to external hub ports, e.g. 1..a, of which each hub port lies on a different backplane segment 1..Q. These other implementations merely require information to be given to the intelligent agent, noting which ports on which of the devices in the agent's hardware environment are connected to the packet-forwarding device(s). That information could be provided by other internal or external software mechanisms (e.g. management application topology information, messaging from a management application, or an automatic switch detection module internal to the agent), or by user input (e.g. configuration via a simple management interface).

In the presently preferred embodiment of the invention, the packet forwarding module 32 is not tightly integrated into the system, in that the intelligent agent cannot control or communicate with the module's switching fabric (e.g. packet-forwarding or address table) hardware. Thus, the herein-disclosed automatic load-balancing method and apparatus preferably accommodates the inability of a system user or the intelligent network agent to communicate with the packet-forwarding support hardware. This feature of the invention provides at least one major advantage over prior art approaches in that the invention is also usable with switches and other packet-forwarding devices which are fully external to the intelligent agent's hardware environment (including switches, bridges, or routers made by other vendors), or which are not manageable by users, such as network administrators. The invention may also be used with an external software application rather than inside an embedded software agent.

Fig. 3 is a block schematic diagram of an apparatus for automatic load balancing on segmented devices according to the invention. The automatic load-balancing method and apparatus further comprise three major components:

- A decision-making process 40 for automatically detecting when a given load-balancing activation currently would not be beneficial;
- An intelligent load-balancing port-distribution algorithm 41 for assigning ports to segments; and
- An undo prior load-balancing feature 42 responsive to an undo command 43 issued by a user.

For network communications between all nodes to be preserved, a means for forwarding packets between the segments (*e.g.* a bridge, a switch or a router) must be present and enabled for the automatic load-balancing feature to be useful for the user.

#### AUTOMATIC DETERMINATION OF WHEN A GIVEN LOAD-BALANCING ACTIVATION WOULD NOT BE BENEFICIAL

One goal for an intelligent feature of any sort involves determining when to actually implement the feature. The herein disclosed automatic load-balancing invention checks for situations in which a given automatic load-balancing activation should not occur. Such situations include, for example, when the current load distribution is already good enough or when current network resource impact is so low that no real problems exist. This automatic determination of when a given instance of feature activation is not beneficial aspect of the automatic load-balancing invention weighs the potential benefit of the action against its consequences, and makes an

intelligent decision for the user while preventing needless network disruption.

## INTELLIGENT PORT-DISTRIBUTION ALGORITHM

Automatic load-balancing moves end-nodes (or groups of end-nodes) after the system has been running to increase the overall throughput of the network. This aspect of the invention redistributes hub ports to segments with the goal of evenly balancing resource load across all of the known segments. Past history is used to anticipate the future load, *i.e.* history since either activation of the intelligent agent or since the last time the load-balancing feature itself was activated. The history can consist of the utilization on each port, communications patterns on the port, communications patterns on the segments, HP EASE data, RMON data, sampling data, or other intelligent considerations. Using the past history, this aspect of the invention moves ports to different segments to try to achieve the same network resource impact on all segments as much as possible or necessary. It should be appreciated that the load-balancing features in general could be applied to any number of segments, although four segments are used in the preferred embodiment of the invention. Additionally, the load-balancing features could also be implemented by an external software application rather than the embedded intelligent agent described in this embodiment.

A number of intelligent decisions, including which ports are to be moved, are incorporated into achieving a new load-balancing distribution. Many of these decisions further the goal of minimizing the number of ports moved from their current segment to a new segment. This is because such an action may cause a temporary disruption to any conversations which may be occurring between one or more of the moved end-nodes. These conversations occur at Open Systems Interconnection (OSI) layers higher than those with which hubs and switches concern themselves. The provision of a good distribution while minimizing the unnecessary moving of ports is one unique feature of the invention. The intelligent port distribution algorithm strives to achieve a

very good network resource distribution across all segments while moving as few ports as possible. This feature of the invention includes assuring that all of the top resource-impacting end-nodes, *i.e.* the greatest users of the available network resources, are distributed across different segments. In addition, it accommodates situations where a particular segment may not be a good location upon which to place a particular port, *i.e.* it may not be fully qualified (*e.g.* when that segment is isolated from the multi-segment backplane on the device on which the port resides), and situations where the port should not be moved (*e.g.* when an external packet-forwarding device is connected to the port, or when a user or management application has designated the port not to be moved).

The herein disclosed algorithm first collects a snapshot of information about all of the ports to be considered for redistribution (automatically excluding ports which should not be moved). This information could include many network resource attributes (*e.g.* utilization, packets, broadcast/multicast packets, collisions, late collisions, etc.), but the presently preferred embodiment minimally notes:

- Management repeater port identification value;
- Management repeater previous segment identification value; and
- Amount of port network resource impact since power-up or last load-balancing activation.

The preferred embodiment of the invention sorts this list of ports in descending order of network resource impact value. A list of segments on which to place the ports is also created. The algorithm proceeds, beginning with the port with the greatest network resource impact value and processing the ports in descending order. The goal is to try to keep as many

ports as possible on their previous segments in an intelligent fashion, while still obtaining a good overall load-distribution.

Two such key areas of the current implementation are:

- Detecting when further changes in port-to-segment assignments are not of sufficient benefit; and
- Finding a well-chosen replacement port-segment assignment (if any) to use in place of each obvious pairing which the basic well-known selection algorithm (the "bin-packing" algorithm) would make.

The replacement assignments in the latter case must be in range of the basic selection algorithm's obvious port-segment assignment. For purpose of the discussion herein, in range means that the AlternatePort or AlternateSegment has some current decision-trigger value (for instance, individual port network resource impact value or segment total assigned network resource impact value) that is some deviationValue distance from the CandidatePort's or TargetSegment's value. The deviationValue could be a fixed value, a fixed percentage, the CandidatePort's percentage of the total network resource impact on all segments, or some other intelligent consideration.

Fig. 4 is a flow diagram showing a preferred implementation of an intelligent port distribution algorithm according to the invention. The following pseudo code listing explains the algorithm in connection with Fig. 4.

while (there are still ports to process) (100; 102)

{

sorted  
by this

Find the next candidate port (104). This is the next port in the list which has not already been marked as assigned algorithm.

Sort the segments in ascending order (106) of current total of the network resource impact of all ports placed on them by the algorithm.

impact  
the  
from one  
that there  
given port on the

Find the first segment with the least current network resource total that is fully qualified as the target segment for CandidatePort (the restrictions for fully qualified vary implementation to another, but essentially indicate are no known adverse effects to putting the given segment).

Make this the TargetSegment (108).

Check if the CandidatePort was previously on the TargetSegment (110).

If so, assign it back to that same segment (124) and loop again above to process any further ports (122, 102).

Check whether the CandidatePort's network resource impact value is below a certain low threshold value, the SmallImpact threshold (112).

If so, check whether the total network resource impact of all remaining unplaced ports that previously were on the CandidatePort's same segment is below another certain low threshold value (126), the MaxAcceptableContribLevel threshold.

If so, further changes in segment assignments for these particular ports is not of sufficient benefit. Assign the CandidatePort and all of those other ports back to their previous segment (128) (this prevents moving small impacters who are not really affecting that much of the network's resources). Then loop again above to continue processing the ports on other segments (122, 102).

If not, continue processing the CandidatePort.

Check whether an AlternateSegment exists for the CandidatePort (116).



If the segment the CandidatePort was previously on is fully qualified with regard to the CandidatePort and is in range of the TargetSegment, an AlternateSegment exists. Assign the CandidatePort to this AlternateSegment (132) and loop again (122, 102) above to process any further ports.

If none of the exceptions above were true, the TargetSegment is used.

Look for an AlternatePort to assign to this TargetSegment in place of the CandidatePort. Proceeding in order through the list of sorted ports, check whether there is as yet a port not already placed by this algorithm that is in range to be used as an AlternatePort and for which the TargetSegment is fully qualified for placement (118).

If an AlternatePort exists within range, assign it to the TargetSegment instead of the CandidatePort (134). Loop above again (122, 102), where the next CandidatePort is the same port as for this loop.

If no AlternatePort exists within range, assign the CandidatePort to the TargetSegment (120) and loop above again (122, 102) to process any further ports.

}

The use of intelligent checking for opportunities to assign ports back to the segments they were previously on minimizes needlessly moving ports, while at the same time obtaining a very good load for each segment. These intelligent checks are unique enhancements to the basic, well-known "bin-packing" selection algorithm.

## UNDO PRIOR LOAD-BALANCING

With any user interface, there is the possibility of selecting and activating a feature by mistake. This is true whether the user interface is a simple text console or a more sophisticated mouse-driven Graphical User Interface (GUI).

If a user were to accidentally activate the automatic load-balancing feature, it would be very hard for them to later undo their mistake. This is because the same embedded intelligence that makes smart decisions on the user's behalf also hides information to keep from overwhelming the unsophisticated user. Even if the user did know the exact full port-to-segment configuration prior to activating the automatic load-balancing feature, it would be tedious for them to return the ports and segments to that configuration themselves. For example, in the current embodiment of the invention, hundreds of ports may be involved.

The undo prior load-balancing feature takes information stored at a prior load-balancing activation and uses it to return ports to that prior configuration. No action is taken on devices which were not known to the agent at the time of that prior load-balancing.

The automatic load-balancing invention offers several advantages, for example as follows:

First, it offers a decision-making process that provides intelligent automatic determination of when a given load-balancing activation currently would not be beneficial, thereby keeping the user's network from being unnecessarily disturbed when no real benefit would be gained. This user-friendly feature is enormously helpful to users who do not fully understand networks and who do not have any real sense of when network conditions do or do not suggest taking an action.

Second, it offers an undo prior load-balancing process when users have accidentally activated a feature which could substantially change their network configurations. This user-friendly feature saves users from their own mistakes, in a situation where it would otherwise be very hard for them to undo such a mistake.

5 The preferred embodiment of the invention provides one level of undo, although other embodiments of the invention could have many levels of undo. Each level of undo takes the user back to a next previous configuration, with the exception that newly added devices are not affected. The invention stores the previous configuration, but alternative embodiments of the invention could also provide the user with choices of default configurations, *e.g.* templates that provide configurations for different functions.

10 Third, it offers an extensive intelligent port-distribution algorithm that improves the network load by moving the right ports to new segments, while at the same time minimizing moving ports unnecessarily. With hundreds of possible ports to choose from in the maximum configuration allowed in the preferred embodiment of the invention, it would be tedious for even a sophisticated user to choose the right ports to move. Unsophisticated users do not have enough knowledge about how a network works to begin to know how to choose whether to move ports. This user-friendly feature benefits users who want better resource availability in their networks, but who lack the knowledge or the inclination to make changes themselves to improve this availability. Combined with the automatic determination of when a given instance of feature activation would not be beneficial, and the undo prior load-balancing features, it also lessens the possibility of users making choices that they may later regret.

25 Moreover, the intelligent port-distribution algorithm described herein merely requires multiple ports, and multiple segments to assign them to. It can be used when the intelligent network software agent does not have access to any packet-forwarding address table information. Although an internal packet-forwarding module is used in this embodiment of the invention, it should be clear that with additional information from other internal or

external software mechanisms, or with minor configuration assistance from the user, other embodiments are possible in which the packet-forwarding device is external to the hub stack and the intelligent agent.

- 5 Although the invention is described herein with reference to the preferred embodiment, one skilled in the art will readily appreciate that other applications may be substituted for those set forth herein without departing from the spirit and scope of the present invention. Accordingly, the invention should only be limited by the Claims included below.